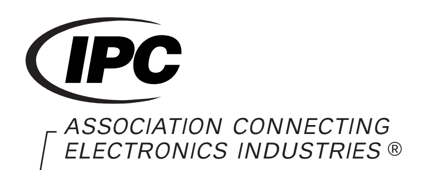


IPC-2501

Definition for Web-Based Exchange of XML Data (Message Broker)

IPC-2501

July 2003

A standard developed by IPC

2215 Sanders Road, Northbrook, IL 60062-6135
Tel. 847.509.9700 Fax 847.509.9798
www.ipc.org

The Principles of Standardization

In May 1995 the IPC's Technical Activities Executive Committee adopted Principles of Standardization as a guiding principle of IPC's standardization efforts.

Standards Should:
- Show relationship to Design for Manufacturability (DFM) and Design for the Environment (DFE)
- Minimize time to market
- Contain simple (simplified) language
- Just include spec information
- Focus on end product performance
- Include a feedback system on use and problems for future improvement

Standards Should Not:
- Inhibit innovation
- Increase time-to-market
- Keep people out
- Increase cycle time
- Tell you how to make something
- Contain anything that cannot be defended with data

Notice

IPC Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such Standards and Publications shall not in any respect preclude any member or nonmember of IPC from manufacturing or selling products not conforming to such Standards and Publication, nor shall the existence of such Standards and Publications preclude their voluntary use by those other than IPC members, whether the standard is to be used either domestically or internationally.

Recommended Standards and Publications are adopted by IPC without regard to whether their adoption may involve patents on articles, materials, or processes. By such action, IPC does not assume any liability to any patent owner, nor do they assume any obligation whatever to parties adopting the Recommended Standard or Publication. Users are also wholly responsible for protecting themselves against all claims of liabilities for patent infringement.

IPC Position Statement on Specification Revision Change

It is the position of IPC's Technical Activities Executive Committee (TAEC) that the use and implementation of IPC publications is voluntary and is part of a relationship entered into by customer and supplier. When an IPC publication is updated and a new revision is published, it is the opinion of the TAEC that the use of the new revision as part of an existing relationship is not automatic unless required by the contract. The TAEC recommends the use of the latest revision.

Adopted October 6. 1998

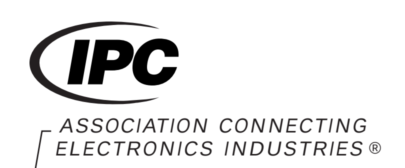
Why is there a charge for this document?

Your purchase of this document contributes to the ongoing development of new and updated industry standards and publications. Standards allow manufacturers, customers, and suppliers to understand one another better. Standards allow manufacturers greater efficiencies when they can set up their processes to meet industry standards, allowing them to offer their customers lower costs.

IPC spends hundreds of thousands of dollars annually to support IPC's volunteers in the standards and publications development process. There are many rounds of drafts sent out for review and the committees spend hundreds of hours in review and development. IPC's staff attends and participates in committee activities, typesets and circulates document drafts, and follows all necessary procedures to qualify for ANSI approval.

IPC's membership dues have been kept low to allow as many companies as possible to participate. Therefore, the standards and publications revenue is necessary to complement dues revenue. The price schedule offers a 50% discount to IPC members. If your company buys IPC standards and publications, why not take advantage of this and the many other benefits of IPC membership as well? For more information on membership in IPC, please visit www.ipc.org or call 847/790-5372.

Thank you for your continued support.



IPC-2501

Message Broker
– GENERIC

Definition for Web-Based Exchange of XML Data

A standard developed by the CAMX Frameworks Communication Committee (2-50).

The IPC-2501 standard specifies the governing semantics and an XML based syntax for shop floor communication between electronic assembly equipment and associated software applications. Wherever possible, existing and widely accepted protocols have been utilized. Certain guaranteed behaviours have been defined to ensure that mission-critical data is reliably communicated among Clients. The purpose of the standard is to outline the communication architecture, supporting XML messages, and to define the choreography between sender and receiver.

Users of this publication are encouraged to participate in the development of future revisions.

Contact:

IPC
2215 Sanders Road
Northbrook, Illinois
60062-6135
Tel 847 509.9700
Fax 847 509.9798

Acknowledgment

Any document involving a complex technology draws material from a vast number of sources. While the principal members of the CAMX Frameworks Communication Committee (2-50) are shown below, it is not possible to include all of those who assisted in the evolution of this standard. To each of them, the members of the IPC extend their gratitude.

CAMX Frameworks Communication Committee

Chair
Andrew D. Dugenske
Georgia Institute of Technology

Technical Liaisons of the IPC Board of Directors

Nilesh S. Naik
Eagle Circuits Inc.

Sammy Yi
Flextronics International

CAMX Frameworks Communication Committee

Robert E. Neal, Agilent Technologies

Kay Lannen, Agilent Technologies

Jeremy Nuanes, Agilent Technologies

Jason Schnitzer, Agilent Technologies

John Minchella, Celestica

Robert Voitus, Celestica

Jorge Camargo, Cookson Electronics Equipment Group

Richard Johnson, DEK Printing Machines Ltd.

Andrew Oughton, DEK Printing Machines Ltd.

Mike Rogers, DEK Printing Machines Ltd.

Richard Coblens, Fuji America Corporation

Monte Cramer, Fuji America Corporation

Michael Kimpton, Fuji America Corporation

Kevin Kroplewski, Fuji America Corporation

Tony Picciola, Fuji America Corporation

Andrew D. Dugenske, Georgia Institute of Technology

Douglas A. Furbush, Georgia Institute of Technology

Andrew Scholand, Georgia Institute of Technology

Jeffrey Gerth, Georgia Tech Research Institute

John Cartwright, Intel Corporation

Douglas Jackson, Intel Corporation

David Martin, Intel Corporation

Hannu Ronkainen, Jot Automation

Mark Doyle, Mapics, Inc.

Brian Nigro, Mapics, Inc.

Brent Bohmont, Motorola Inc.

Mark Williams, Motorola

Dan Pattyn, Motorola, Inc.

Jim Brazelton, NACOM Corporation

Louis Watson, NACOM Corporation

Kevin Brady, NIST Nat'l. Institute of Stds & Technology

Barbara Goldstein, NIST Nat'l. Institute of Stds & Technology

Michael McLay, NIST Nat'l. Institute of Stds & Technology

John Messina, NIST Nat'l. Institute of Stds & Technology

Rick Lloyd, Nortel Networks

Dave J. Morris, Nortel Networks

Tony Wong, Nortel Networks

Hitoshi Nakamura, Panasonic

Tom Baggio, Panasonic Factory Automation

Hiro Kurata, Panasonic Factory Automation

Tak Yokoi, Panasonic Factory Automation

Moustafa Noureddine, Router Solutions Inc.

Robert Schwanke, Siemens

Dilip Soni, Siemens

Cord Burmeister, Siemens Dematic Corporation

Tuan M. Nguyen, Siemens Dematic Elect. Asmbly. Systems

Art Sedighi, Talarian Corporation

Grace Yee, Talarian Corporation

Niko Siltala, Tampere University of Technology Institute of Production Engineering

Carey Price, TechCenter

Rudi Streif, Teradyne Inc.

Allan Fraser, Teradyne Inc.

Rob Bryla, Universal Instruments Corporation

Tom J. Dinnel, Universal Instruments Corporation

Jerry Lowery, Visiprise, Inc.

Table of Contents

Definition for Web-Based Exchange of XML Data

Introduction

Information flow is essential to efficient electronics manufacturing and standards are essential to information flow. One area of commerce that has lacked its own communication standards is the electronics manufacturing factory floor. Information exchange between a system of electronic assembly equipment and higher-level applications has, in the past, used proprietary or borrowed standards. The IPC-254X and IPC-255X series of standards address this issue by defining the messages needed for this information exchange.

Just as “snail mail” and e-mail information exchanging requires standards for the envelope and transportation mechanisms – or messaging interface – so also do factory floor communications. This standard describes a messaging interface that is based upon an architecture whereby a single logical middleware server (the Message Broker) exchanges messages among Clients in a Domain.

Clients may be electronics manufacturing equipment or software applications present in the domain. The Message Broker acts as an intelligent message router between these Clients, accomplishing both Point-to-Point and Publish/Subscribe communications. Figure 1 shows a computer-aided manufacturing XML (CAMX) Domain consisting of the Message Broker, Application Clients, and Equipment Clients.

Although the scope of this document is that of electronics manufacturing, the broader applicability of the message transport mechanisms defined by this standard must be noted. Any application that uses XML-based messages can use the IPC-2501 web-based exchange mechanism. Specifically, this standard can support the exchange of XML-based messages both within an enterprise, be it manufacturing or service based -- and externally -- between multiple enterprises having the need for efficient, reliable web-based communications. Broad adoption of this standard is strongly encouraged.

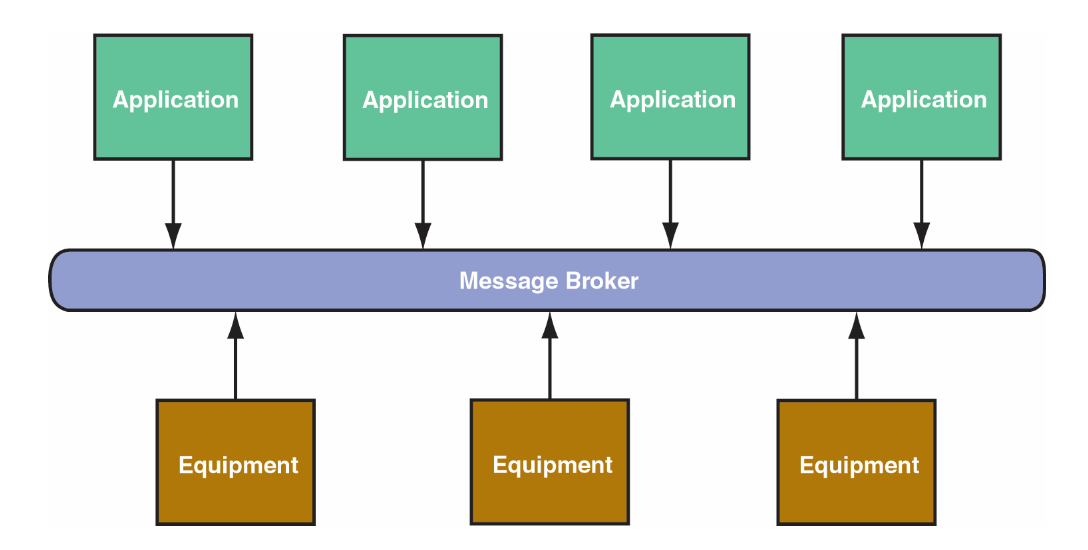


Figure 1 Example IPC-2501 Domain consisting of the Message Broker, Application Clients, and Equipment Clients

1 Scope

The intent of this standard is to establish the governing semantics and an XML based syntax for shop floor communication between electronic assembly equipment and associated software applications. Wherever possible, existing and widely accepted protocols have been utilized. Certain guaranteed behaviors have been defined to ensure that mission-critical data is reliably communicated among Clients.

The purpose of this specification is to outline the communication architecture and supporting XML messages. The required programmatic actions that define the choreography between sender and receiver have also been defined.

The domain of this standard is that of an electronics assembly manufacturing shop, consisting of up to several hundred machines, each of which is capable of producing tens of messages per second. Most of these messages are relatively small in size (under 20 kilobytes); however some application-specific files of several megabytes will occasionally need to be transferred. The number of consumers of this information is assumed to be a relatively small number, typically less than 20, and this number does not directly increase in proportion to the number of machines. Provisions have also been made to accommodate network interruptions.

1.1 General Design Principals

Many different levels of system complexity are possible in addressing the intent outlined above. The industry participants guiding the development of this standard set forth the following design principles:

- Low Cost
- Low Complexity
- Stable
- Deterministic
- Centrally Configured
- Scalable

1.2 Intended Audience

This document is intended for an audience of manufacturing system software developers, computer aided manufacturing application programmers and Information Technology professionals as well as an end user community that includes process engineers and manufacturing specialists.

2 Interpretation

"Shall", the emphatic form of the verb, is used throughout this standard whenever a requirement is intended to express a provision that is mandatory. Deviation from a **shall** requirement is not permitted, and compliance with the XML syntax and semantics **shall** be followed without ambiguity, or the insertion of superfluous information. The words "should" and "may" are used whenever it is necessary to express non-mandatory provisions. "Will" is used to express a declaration of purpose.

To assist the reader, the word **shall** is presented in boldface characters.

3 General Requirements

The XML schemas contained in this document describe the structures for a CAMX data exchange. The document specifies data elements specifically designed to establish the information exchange capabilities as related to the electronics manufacturing factory floor. The XML schemas define the configuration of mandatory and optional elements, as well as mandatory and optional attributes.

3.1 Terms and Definitions

The definitions of all terms used herein **shall** be as specified in IPC-1050, and by the following:

Client – A generic term for any of the various machines, applications, or devices that may connect to a Message Broker in a Domain.

Domain - The set of all Clients interested in communicating with each other. It contains a single logical Message Broker and a configurable number of Clients.

Domain Configuration - The Domain Configuration defines publishing capabilities, subscription interests, point-to-point communication privileges, quality of service parameters and general information about the Domain.

Message Broker - The Message Broker is the messaging middleware. It is responsible for intelligently routing messages among Clients.

3.2 Communication Architecture

In accordance with the intent of using broadly accepted standards where possible, this system makes use of TCP/IP, HTTP, XML, and SOAP as illustrated in Figure 2.

Message Exchange	**(IPC 2501)**
Message Semantics	**(IPC 251X - 258X)**
Message Syntax	**(XML)**
Packaging	**(SOAP w/ Attachments)** **(IPC 2501 Extensions)**
Transport	**(HTTP)**
Network	**(TCP/IP)**

Figure 2 IPC-2501 Layered Communications Architecture

There are two chief advantages of this standards-based approach. First, re-invention and re-implementation of basic commodity functionality is minimized. Second, the use of widely accepted Internet standards will make the IPC-2501 more attractive to potential implementers.

The way in which SOAP with attachments and the HTTP protocol are used together to represent IPC-2501 messages is illustrated in Figure 3.

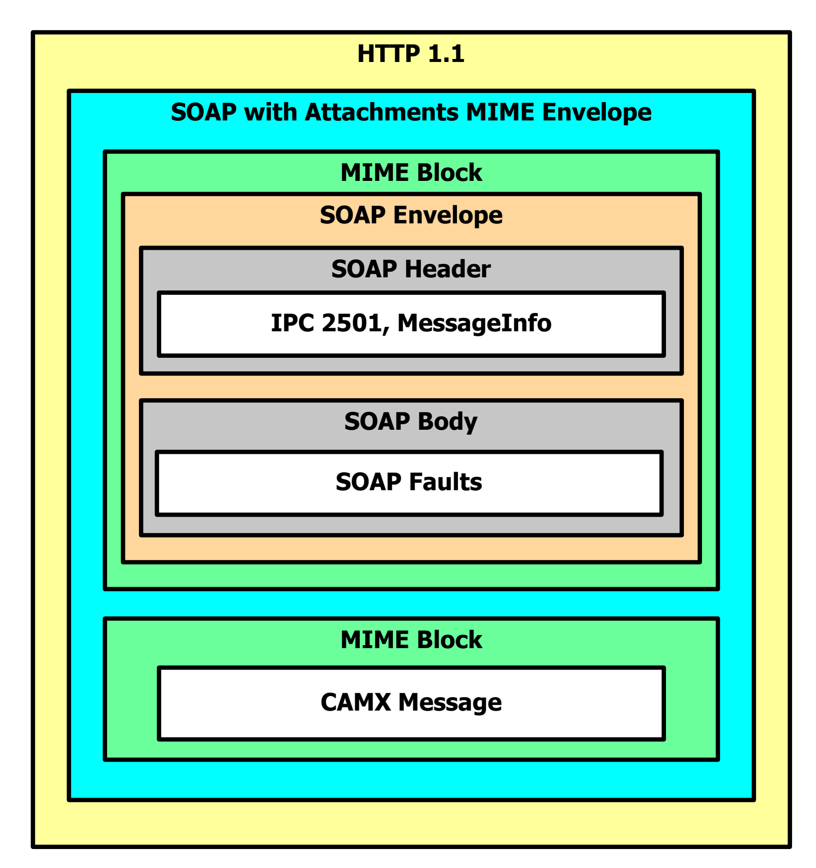


Figure 3 The IPC-2501 transmission structure

3.2.1 TCP/IP Usage

TCP/IP is used by this standard because it is absolutely standardized and offers the highest assurance that all types of systems from all vendors can communicate effectively.

3.2.2 HTTP Usage

All transfers of messages in an IPC-2501 Domain are accomplished via the HTTP 1.1 protocol through the Message Broker. The Message Broker acts as an HTTP Server and the Clients act as HTTP Clients.

A transaction consists of two HTTP transmissions. The first transmission is an HTTP Client Request and **shall** be initiated by a Client. The second transmission is an HTTP Server Response, and **shall** be accomplished by the Message Broker (see Figure 4).

The HTTP Client Request **shall** use the HTTP POST method. The HTTP Server **shall** Respond with either an HTTP 200 or an HTTP 500 status code. An HTTP 200 status code **shall** indicate that the HTTP request has succeeded. An HTTP 500 status code **shall** indicate the Message Broker encountered an unexpected HTTP condition that prevented it from fulfilling the request.

For further information about HTTP 1.1, see:

http://www.w3.org/Protocols/rfc2616/rfc2616.html.

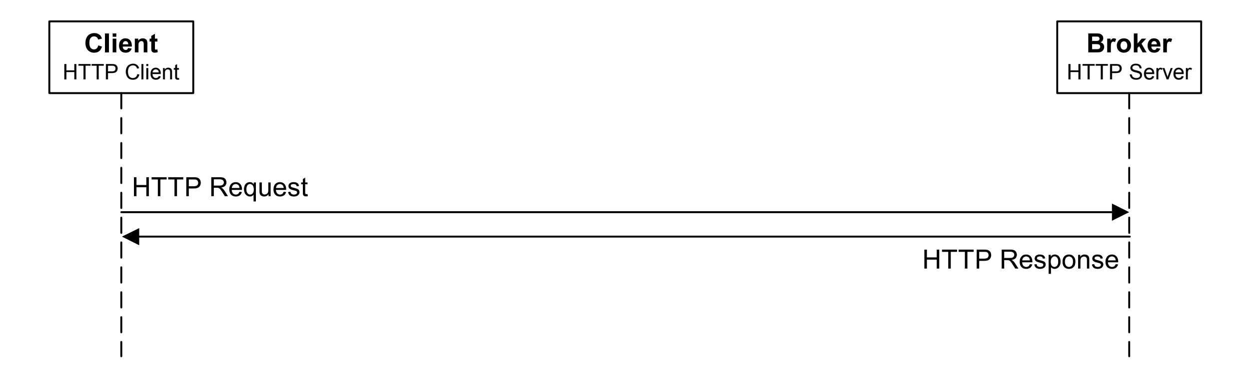


Figure 4 An HTTP Request/Response Transaction between a Client and the Message Broker. All communication in an IPC-2501 Domain follows this pattern.

3.2.3 SOAP with Attachments Usage

In addition to HTTP, SOAP with Attachments **shall** be used to transfer messages. Each HTTP transmission as defined in the previous section **shall** contain two MIME blocks. The first MIME block **shall** contain a SOAP Envelope. The second MIME block **shall** contain the message that is being transferred. (note: Future versions of this standard may use multiple MIME blocks to transfer additional data.)

The SOAP Envelope contains a SOAP Header element and a SOAP Body element. The SOAP Header element **shall** contain an IPC-2501 MessageInfo as a child element. If a SOAP fault has been generated, the SOAP Body **shall** contain the SOAP fault (see Figure 5).



Figure 5 Location of MessageInfo element within the SOAP Envelope

For more information about SOAP Version 1.1, see:
http://www.w3.org/TR/SOAP/

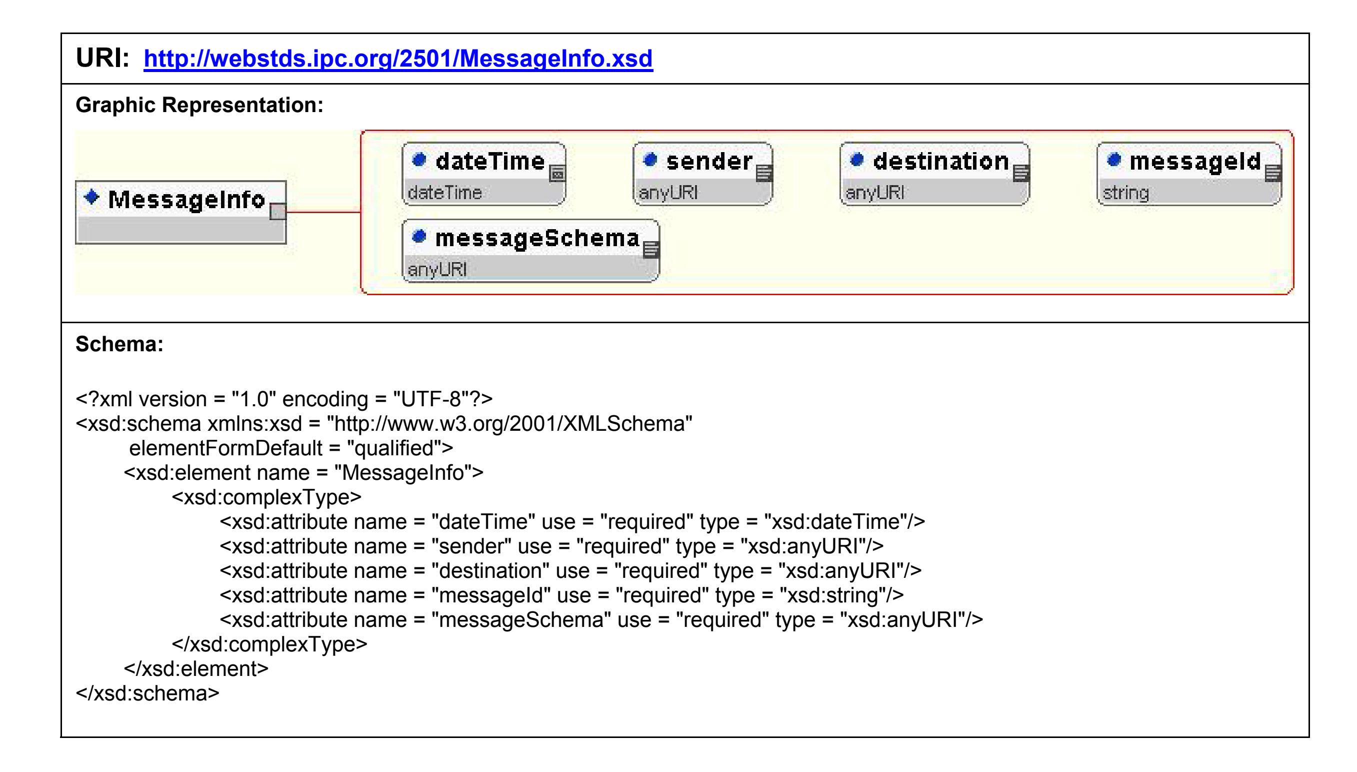
For more information about SOAP with Attachments, see:
http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211.html

The SOAP 1.1 envelope schema can be found at:
http://schemas.xmlsoap.org/soap/envelope/

3.2.3.1 MessageInfo Element

The MessageInfo element contains meta-level information about the message that is contained in the second MIME block of each HTTP transmission. Attributes of the MessageInfo element provide origination and destination context to the message so that it can be routed appropriately. Attributes are also included that indicate the schema of the message, a unique identifier, and the date and time of the message.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
sender	anyURI	The unique, predefined name of the entity in the Domain sending this message.	1
destination	anyURI	The unique, predefined name of the entity in the Domain where this message is to be sent.	1
dateTime	dateTime	The date and time of the message.	1
messageSchema	anyURI	The location of the schema of the attached message represented by this element.	1
messageId	string	An attribute representing the unique identifier of the message.	1



URI: http://webstds.ipc.org/2501/MessageInfo.xsd

Graphic Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
     elementFormDefault = "qualified">
    <xsd:element name = "MessageInfo">
        <xsd:complexType>
            <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
            <xsd:attribute name = "sender" use = "required" type = "xsd:anyURI"/>
            <xsd:attribute name = "destination" use = "required" type = "xsd:anyURI"/>
            <xsd:attribute name = "messageId" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "messageSchema" use = "required" type = "xsd:anyURI"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

3.2.3.1.1 Unique ID Generation

Each message **shall** be uniquely identified for the purpose of acknowledgement, logging, response correlation and other purposes. Identifier strings must be printable characters and may be generated in any fashion that will produce an identifier with a name space collision possibility of zero.

Recommended Methods:

Systematic identification can be attained through requests for a 128 bit Globally Unique Identifier (GUID) or Universally Unique Identifier (UUID). Where such identification is not readily available the recommended methodology is to concatenate the sender's internet protocol (IP) address or networking hardware machine (MAC) address with a string representing the current date, time and time zone. This will typically produce 1/100 second accuracy. Where this is not of sufficient resolution to meet the message generation rate, appending an additional counting sequence is recommended (i.e. "130.207.198.100|20010501013725.85+5000|00").

3.3 Message Transfer

Most all CAMX messages will be exchanged among two or more Clients through the Message Broker. Only messages conveying Domain information will involve just the Message Broker and an individual Client. All messages, including those from one Client to another, **shall** be sent to the Message Broker. In this respect the Message Broker functions as a publisher, taking a submission from a Client and distributing it to all the Clients that have requested submissions of that type. Unsolicited submissions are not allowed; the Message Broker obtains a list of messages each Client can generate and makes the list of messages available to all Clients. The Message Broker then instructs each Client to generate only those messages to which at least one Client has subscribed.

Under normal operations a Client needs only to send a message once, regardless of the number of subscribers. It is the responsibility of the Message Broker to ensure the message is replicated and delivered to each and every Client who has subscribed to that message. Consuming Clients may receive the messages not in chronological sequence though.

When the Message Broker receives a submitted message it places the message in a first-in-first-out queue and awaits a request for messages from the subscribed Client(s). Each subscribed Client periodically queries the Message Broker as to whether or not there are any queued messages (see 4.4 GetMessage). By this practice, messages will be received by the Client in the sequence that they were received by the Message Broker.

The normal rules here established for publication of messages do not apply to point-to-point communication. A Client, if authorized in the Domain configuration, can send a message to another Client. Such a message might, for example, request information as to the current value of a specific parameter. Such point-to-point messages **shall** follow the same rules as published messages in that they are sent to the Message Broker who queues the message, completing the delivery only when the receiving Client requests its messages. Sections 3.3.1 to 3.3.4 contain additional details about successfully transferring a message. Section 5 contains guidance for handling errors that might occur when attempting to transfer a message.

3.3.1 Client to Message Broker Transfer

The transfer of a message from a Client to the Message Broker is accomplished via two HTTP transmissions (see Figure 6).

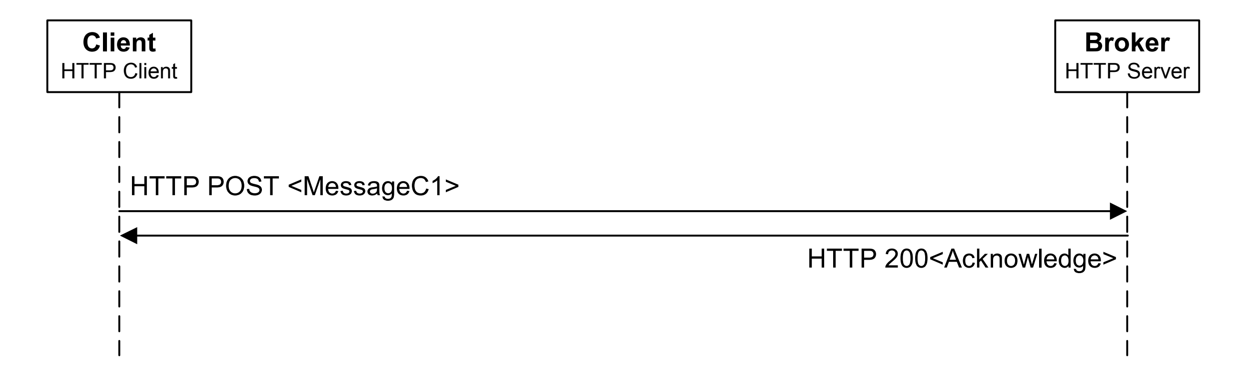


Figure 6 The successful transfer of a message from a Client to the Message Broker. The message was Acknowledged by the Message Broker.

The specifics of the transmissions are as follows:

1. The Client **shall** transmit the message via an HTTP POST request, in which the message to be transferred is included in the second MIME block.
2. The Message Broker **shall** respond with an HTTP 200 status code, and an Acknowledge contained in the second MIME block.

3.3.2 Message Broker to Client Transfer

The transfer of a message from the Message Broker to a Client is accomplished via four HTTP transmissions (see Figure 7).

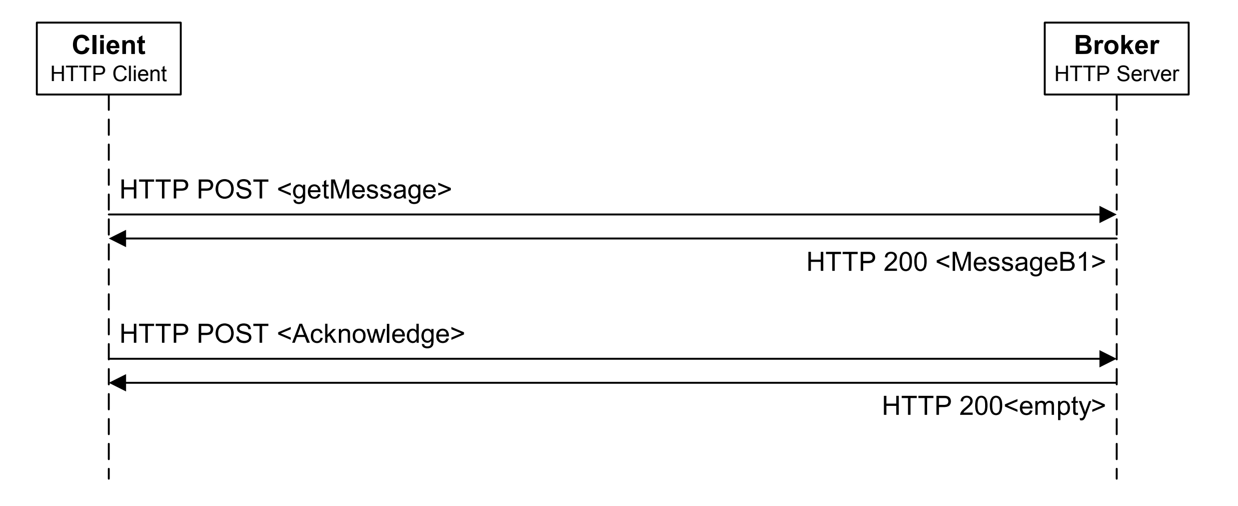


Figure 7 The successful transfer of a message from the Message Broker to a Client

The specifics of the transmissions are as follows:

1. The Client **shall** transmit an IPC-2501 GetMessage via an HTTP POST request, in which the GetMessage is included in the second MIME block.
2. The Message Broker **shall** respond with an HTTP 200 status code, and the message to be transferred contained in the second MIME block.
3. The Client **shall** transmit an Acknowledge via an HTTP POST request, in which the Acknowledge is contained in the second MIME block.
4. The Message Broker **shall** respond with an HTTP 200 status code, and an empty SOAP envelope.

If no messages are queued by the Message Broker for the Client, only two HTTP transmissions will take place (see Figure 8).

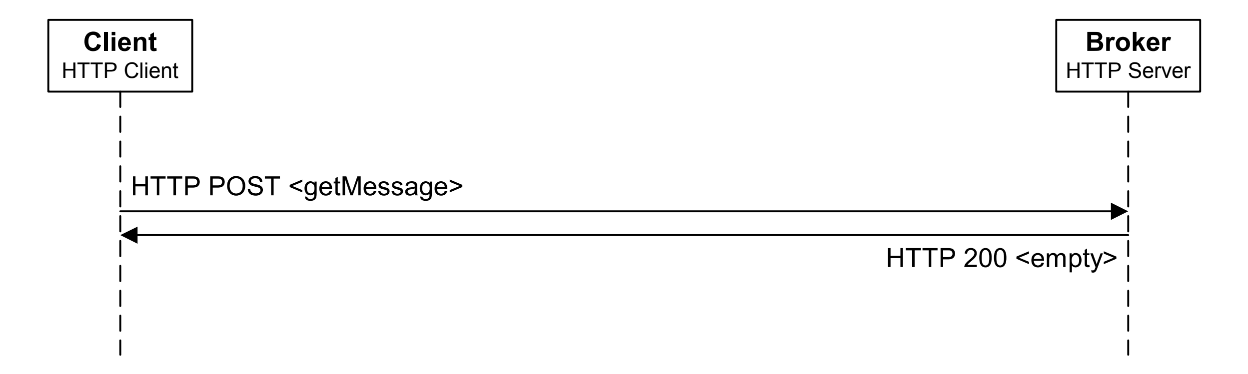


Figure 8 When a message is not waiting for a Client, only two HTTP transmissions take place

The specifics of the transmissions are as follows:

1. The Client **shall** transmit an IPC-2501 GetMessage via an HTTP POST request, in which the GetMessage is included in the second MIME block.
2. The Message Broker **shall** respond with an HTTP 200 status code, and an empty SOAP envelope.

3.3.3 Client to Client Transfer (Point-to-Point)

The transfer of a message from one Client to another Client is accomplished through the Message Broker via six HTTP transmissions (see Figure 9).

Note: This Standard enforces single-level, Broker–Client acknowledgement for Point-to-Point messages; End-to-End acknowledgement is not overtly supported.

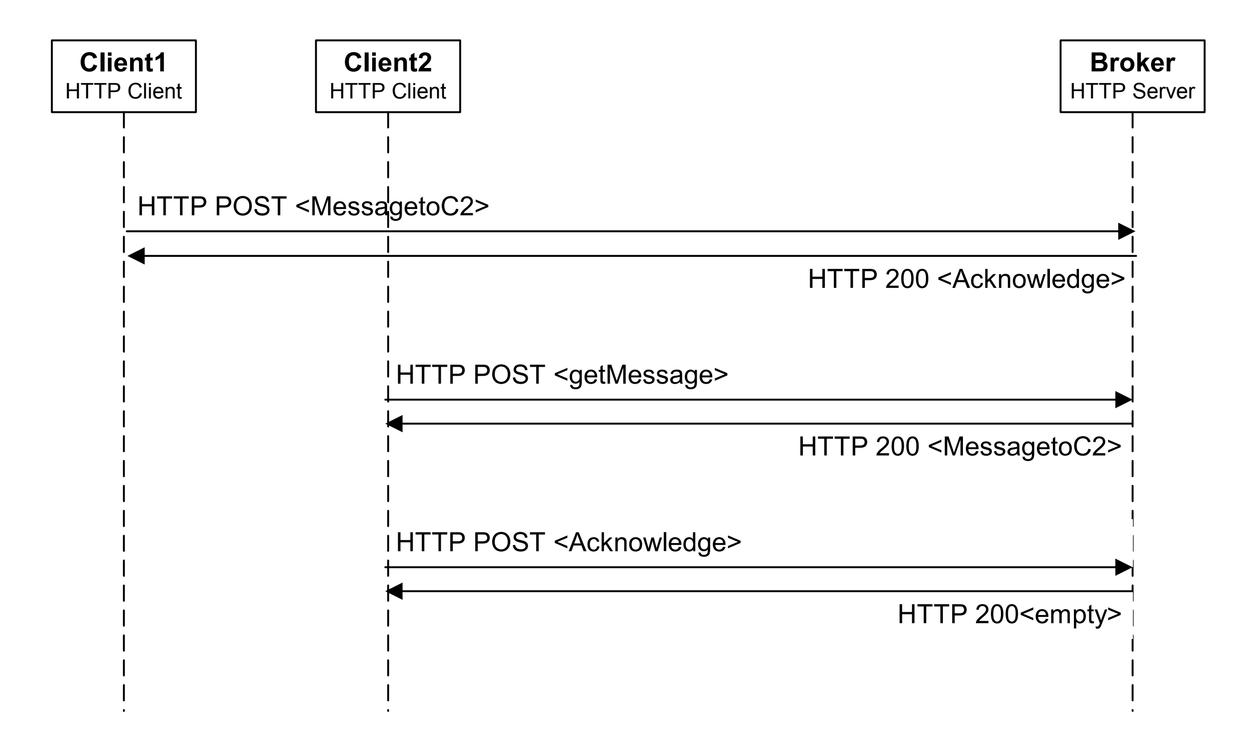


Figure 9 Client to Client transfer

The specifics of the transmissions are as follows:

1. Client1 **shall** transmit the message via an HTTP POST request, in which the message to be transferred is included in the second MIME block.
2. The Message Broker **shall** respond with an HTTP 200 status code, and an Acknowledge contained in the second MIME block.
3. Client2 **shall** transmit an IPC-2501 GetMessage via an HTTP POST request, in which the GetMessage is included in the second MIME block.
4. The Message Broker **shall** respond with an HTTP 200 status code, and the message to be transferred contained in the second MIME block.
5. Client2 **shall** transmit an Acknowledge via an HTTP POST request, in which the Acknowledge is contained in the second MIME block.
6. The Message Broker **shall** respond with an HTTP 200 status code, and an empty SOAP envelope.

3.4 Quality of Service

In the factory floor environment of this standard, the data about the product can be as important as the product itself. Because of this, the industry participants guiding the development of this standard set forth a Quality of Service guideline resulting in the choreography of message queuing, sending, resending and acknowledgement that follows.

3.4.1 Guaranteed Message Delivery

Guaranteed Message Delivery is accomplished by adopting some simple rules aimed at data integrity. The Acknowledge message **shall** be used to indicate when conformance to these rules has been achieved. Because the unlimited application of these data integrity rules could quickly overwhelm the message storage capabilities of the nodes within the IPC-2501 domain, Quality of

Service terms (data integrity rules) can be applied to bound the message storage used by IPC-2501 data for both the Client and the Message Broker. Each of these concepts is described in greater detail in the sub-sections below.

3.4.2 Data Integrity

Data to be published is initially the responsibility of the producing Client and remains its responsibility until such time as it sends the data as an IPC-2501 message AND receives an Acknowledgement of receipt of the message from the Message Broker. Only at that time has the Client passed responsibility for the data exchange to the Message Broker. Until that time a copy of the data **shall** remain with the Client in non-volatile media.

The Message Broker **shall** be responsible for the message data until such time as the Message Broker has passed a copy of the message to all Clients that have subscribed to that message type from the producing Client, AND received an Acknowledgement of receipt of the message from each of these Clients. Until that time a copy of the data **shall** remain with the Message Broker in non-volatile media.

3.4.3 Acknowledge

Acknowledge is a response from either the Message Broker to a Client or vice-versa indicating receipt and acceptance of data.

A Client **SHALL NOT** transmit a new message until the Message Broker has acknowledged the previous message. The Message Broker **SHALL NOT** transmit a new message to a particular Client until that Client has acknowledged the previous message.

Acknowledge specifically **does not** indicate application level cognizance of the message. It does not provide any assurance that the message payload was syntactically correct, was understood, or was agreed to by the recipient. Message Broker Acknowledge does not imply that the message has reached its intended recipient. All messages **shall** be acknowledged with the following exceptions:

1) **Acknowledge** messages themselves are not acknowledged
2) **Error** messages are not acknowledged
3) The empty response (i.e. “no messages queued”) to a **GetMessage** request is not acknowledged

3.4.4 Queue Full Operation

Since media is finite, the non-volatile memory allocated by the Message Broker for a Client may become full. There are two permissible policies for storage overflow: *loss* and *loss-less*. These policies are indicated respectively by ERASE and STOP values in the queueFullOperation attribute of the DomainConfiguration. A QueueFull condition is defined as a queue that contains the maximum number of messages that it can store as specified by the queueSizeAttribute of the Client element, see Section 4.2.4.

The loss policy (indicated by an attribute value of ERASE) permits the discarding of sufficient messages from non-volatile memory to resume operations.

The loss-less policy (indicated by an attribute value of STOP) only allows the discard of messages held for a Client via human intervention (including a domain change). If Message Broker persistent media resources are insufficient to store incoming messages, all such incoming messages are refused (i.e., NOT Acknowledged) until such time as confirmed delivery of

currently persisted messages frees up sufficient non-volatile memory to resume server operations.

Note that these rules as described above establish a delivery policy of *at least once*, meaning if a communication between Message Broker and Client is interrupted before a message exchange transaction is complete, then the initiating party will resend the message. If the interruption results in a corruption of the acknowledgement rather than the message transmission, these rules will result in a duplicate copy of the message arriving at the destination. All 2501 Clients must therefore be tolerant to duplicate messages as identified by the messageId attribute.

3.5 Domain Configuration

The Domain is configured “statically,” or more specifically, externally to the IPC-2501 message exchange process by an Administrator. The DomainConfiguration schema listed in Section 4.2 provides a list of what options may be configured.

It is recommended that all Clients begin message transfer within the Domain by sending the Message Broker an initial GetDomainConfiguration message, to assure the Client is communicating with the intended broker in the intended domain. If the Client is not listed in the Domain Configuration, an Error will be generated (see Section 4.6 Error for a complete list of error codes).

4 IPC-2501 Messages

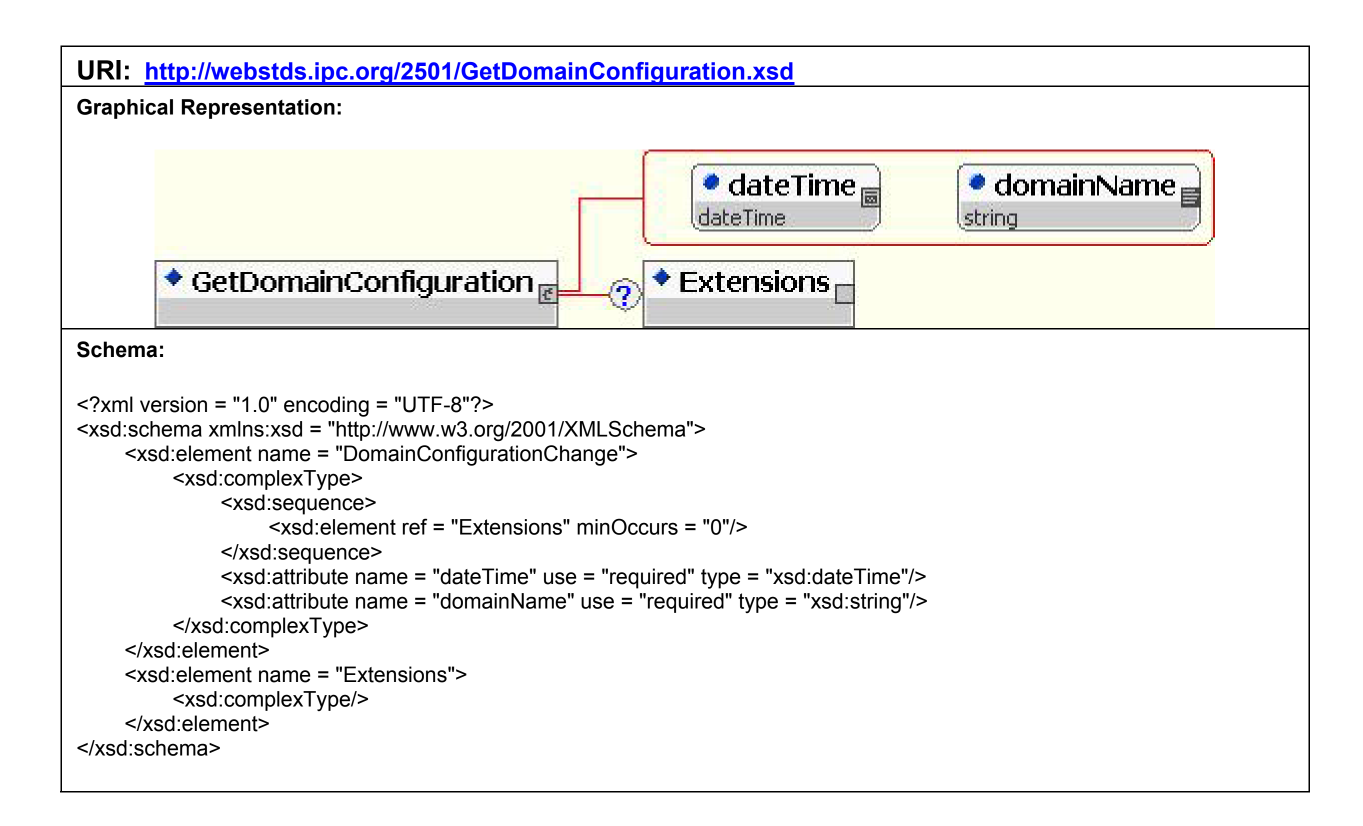
Many functions of the 2501 Domain (such as event notification, persistence, and status retrieval) are controlled via XML messages carried as MIME attachments to the SOAP envelope. The following sections document the information models (schemas) of these messages.

4.1 GetDomainConfiguration

This message is sent by a Client to the Message Broker to obtain the current DomainConfiguration.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
dateTime	dateTime	The time stamp capturing the instant the Client created the GetDomainCharacteristics message.	1
domainName	string	The name of the Domain that the Message Broker is servicing.	1
Extensions	Element	An optional element for containing non-standard XML messages and references.	0-1

[1]Occurrence



URI: http://webstds.ipc.org/2501/GetDomainConfiguration.xsd

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
	<xsd:element name = "DomainConfigurationChange">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref = "Extensions" minOccurs = "0"/>
			</xsd:sequence>
			<xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
			<xsd:attribute name = "domainName" use = "required" type = "xsd:string"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name = "Extensions">
		<xsd:complexType/>
	</xsd:element>
</xsd:schema>
```

4.2 DomainConfiguration

The DomainConfiguration defines the configuration of the Domain. It specifically defines the following:

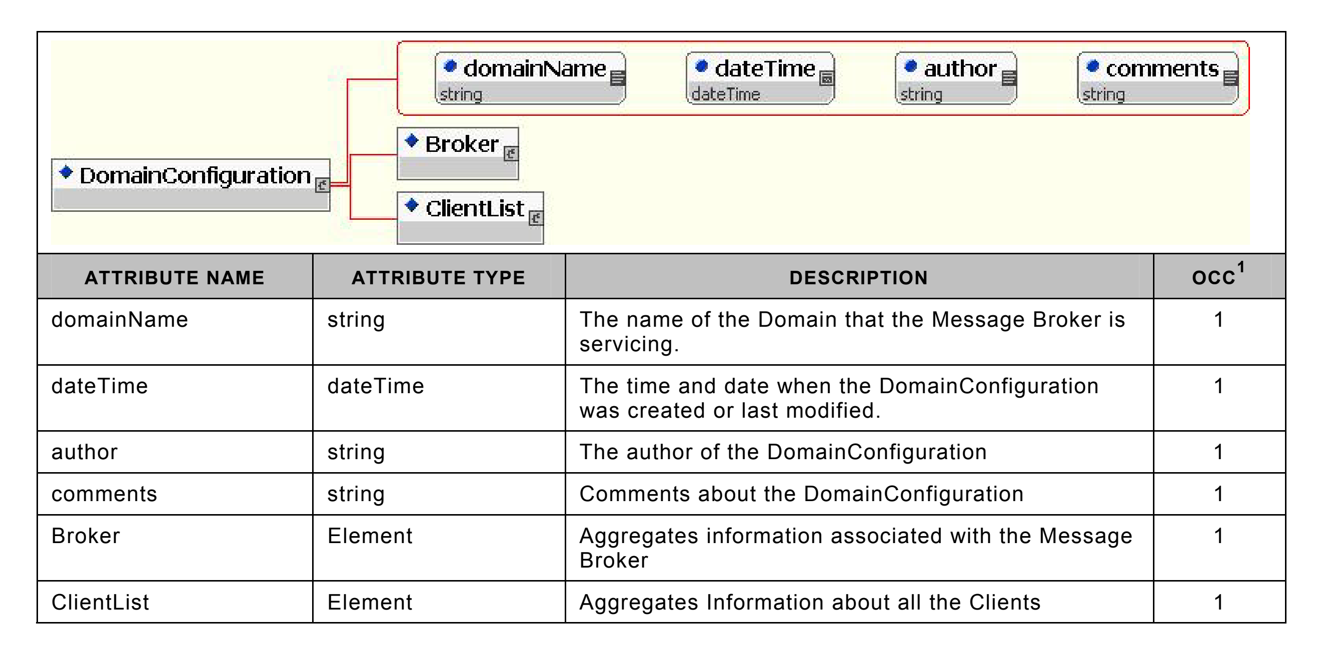
- The publishing capabilities of all Clients and the Message Broker
- The subscription interests of each Client
- The point-to-point messages that the Message Broker will deliver to each Client
- Quality of service information
- General information about the Domain

The DomainConfiguration is returned to a Client in response to a GetDomainConfiguration message.

Individual elements of the schema are described in Sections 4.2.1 to 4.2.7. The comprehensive DomainConfiguration schema is listed in the Section 4.2.8.

4.2.1 DomainConfiguration Element

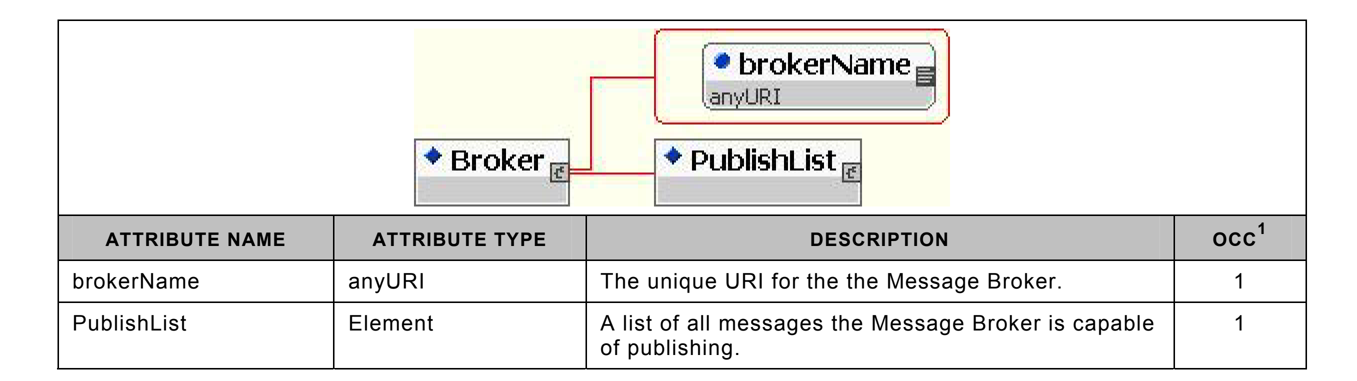
The root element of the DomainConfiguration schema is the DomainConfiguration element. Information about the Message Broker is contained in the Broker sub-element and information about the Clients is contained in the ClientList sub-element. The attributes contain general information about the domain.



ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
domainName	string	The name of the Domain that the Message Broker is servicing.	1
dateTime	dateTime	The time and date when the DomainConfiguration was created or last modified.	1
author	string	The author of the DomainConfiguration	1
comments	string	Comments about the DomainConfiguration	1
Broker	Element	Aggregates information associated with the Message Broker	1
ClientList	Element	Aggregates Information about all the Clients	1

4.2.2 Broker Element

The Broker Element contains information about the Message Broker. Specifically, the messages that the Message Broker is capable of publishing are contained in the PublishList sub-element, and the name of the Message Broker is contained in the attributes.



ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
brokerName	anyURI	The unique URI for the the Message Broker.	1
PublishList	Element	A list of all messages the Message Broker is capable of publishing.	1

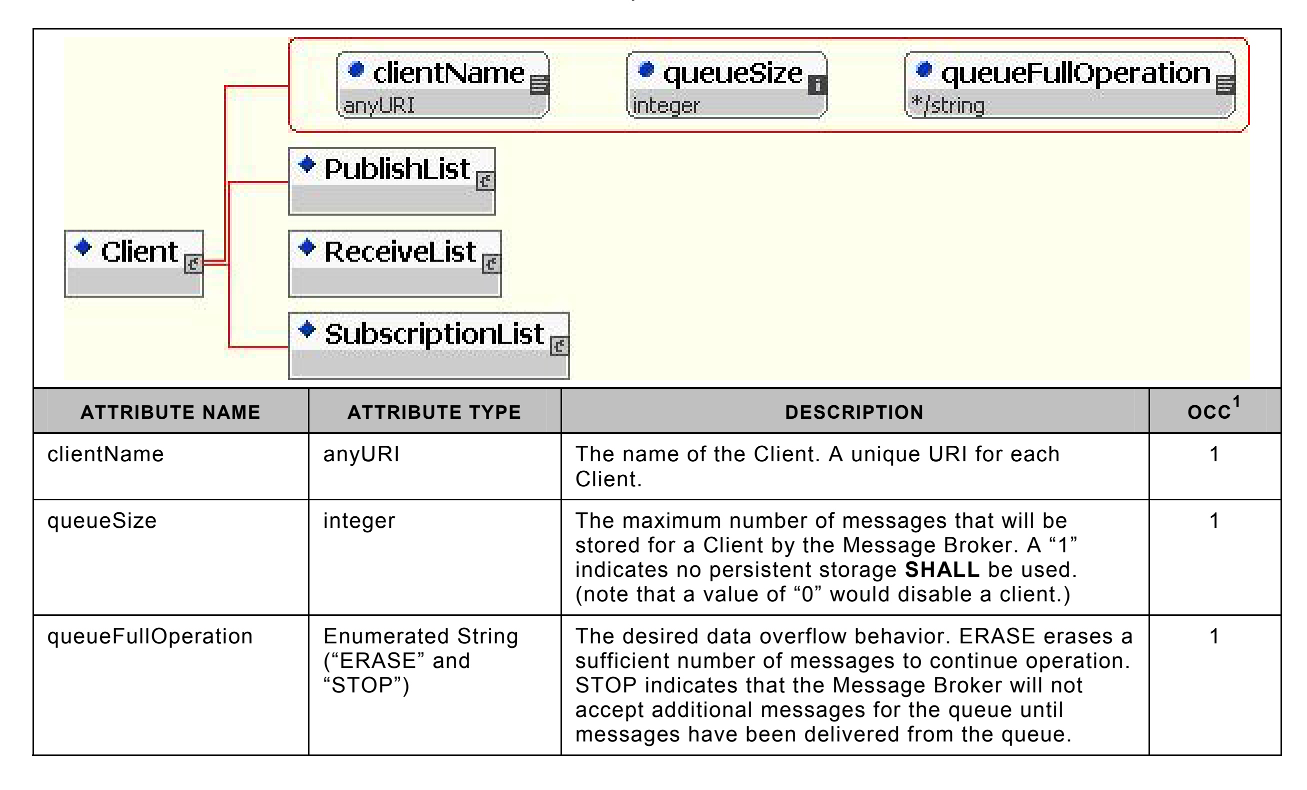
4.2.3 ClientList Element

The ClientList element contains one or more Client elements.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
Client	Element	Aggregates information on a per Client basis	1-n

4.2.4 Client Element

The Client element contains information about an individual Client. Specifically, the messages that the Client is cable of publishing are contained in the PublishList sub-element, the messages the Client will accept are contained in the ReceiveList sub-element and the messages the Client desires to receive are contained in the SubscriptionList sub-element.

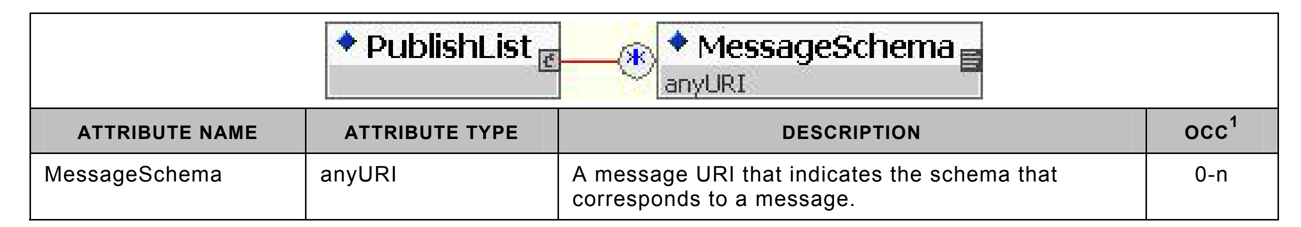


ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
clientName	anyURI	The name of the Client. A unique URI for each Client.	1
queueSize	integer	The maximum number of messages that will be stored for a Client by the Message Broker. A “1” indicates no persistent storage **SHALL** be used. (note that a value of “0” would disable a client.)	1
queueFullOperation	Enumerated String (“ERASE” and “STOP”)	The desired data overflow behavior. ERASE erases a sufficient number of messages to continue operation. STOP indicates that the Message Broker will not accept additional messages for the queue until messages have been delivered from the queue.	1

PublishList	Element	A list of all messages that a Client or the Message Broker is capable of publishing.	1
ReceiveList	Element	The only messages that the Message Broker will send this Client via Point-to-Point communication.	1
SubscriptionList	Element	A list of messages the Client desires.	1

4.2.5 PublishList Element

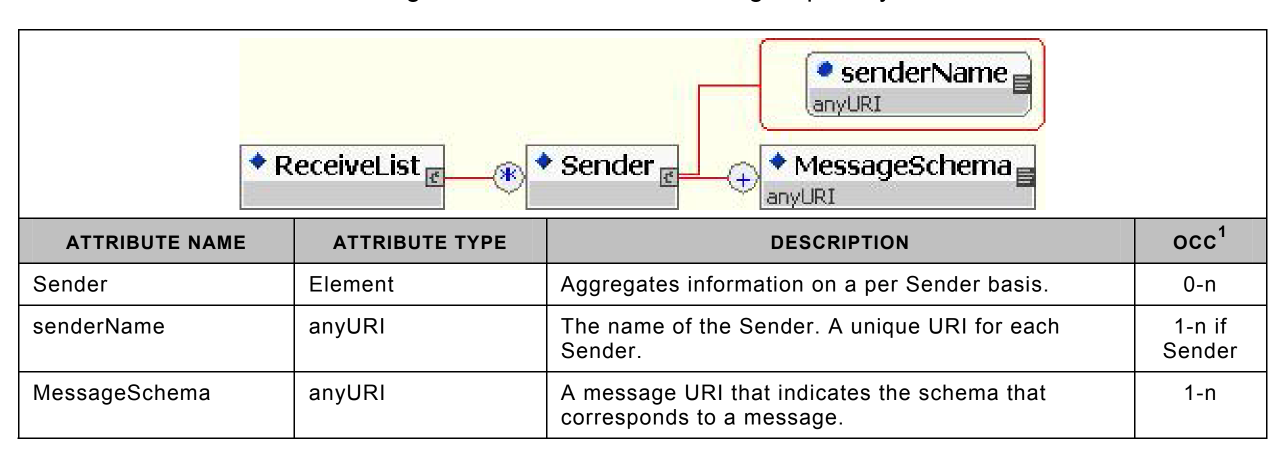
The PublishList element contains the list of messages that the Message Broker or Clients are capable of publishing.



ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
MessageSchema	anyURI	A message URI that indicates the schema that corresponds to a message.	0-n

4.2.6 ReceiveList Element

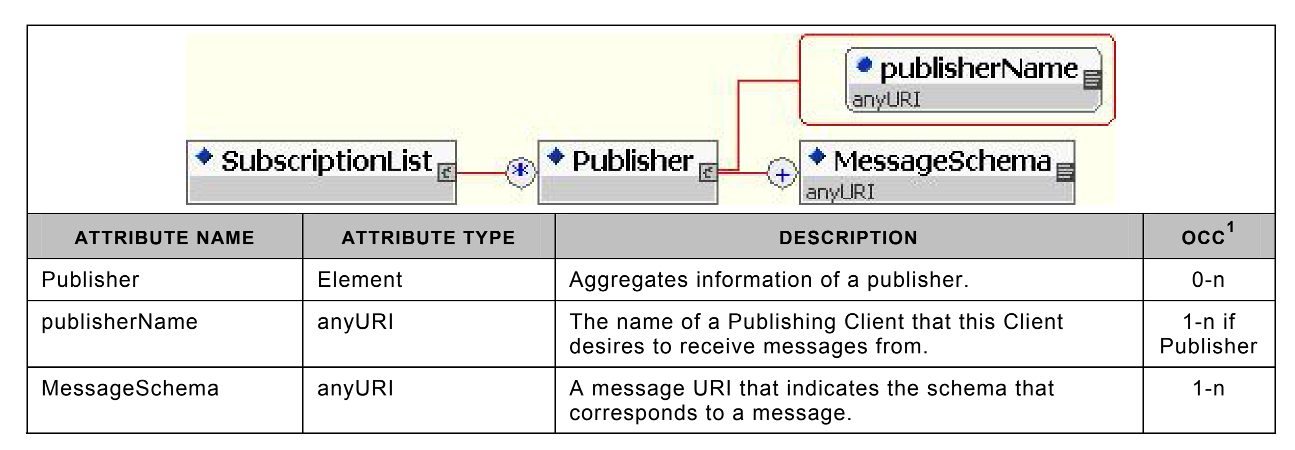
The ReceiveList element contains the list of Point-to-Point messages that a Client will accept from other Clients. The messages in the ReceiveList are grouped by sender.



ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
Sender	Element	Aggregates information on a per Sender basis.	0-n
senderName	anyURI	The name of the Sender. A unique URI for each Sender.	1-n if Sender
MessageSchema	anyURI	A message URI that indicates the schema that corresponds to a message.	1-n

4.2.7 SubscriptionList Element

The SubscriptionList element contains the list of messages that the Client desires to receive from publishing Clients. The messages in the SubscriptionList are grouped by publishing Clients.



ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC[1]
Publisher	Element	Aggregates information of a publisher.	0-n
publisherName	anyURI	The name of a Publishing Client that this Client desires to receive messages from.	1-n if Publisher
MessageSchema	anyURI	A message URI that indicates the schema that corresponds to a message.	1-n

4.2.8 DomainConfiguration

The comprehensive DomainConfiguration schema.

URI: http://webstds.ipc.org/2501/DomainConfiguration.xsd

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "DomainConfiguration">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "Broker"/>
                <xsd:element ref = "ClientList"/>
            </xsd:sequence>
            <xsd:attribute name = "domainName" use = "required" type = "xsd:string"/>
            <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
            <xsd:attribute name = "author" use = "required" type = "xsd:string"/>
```

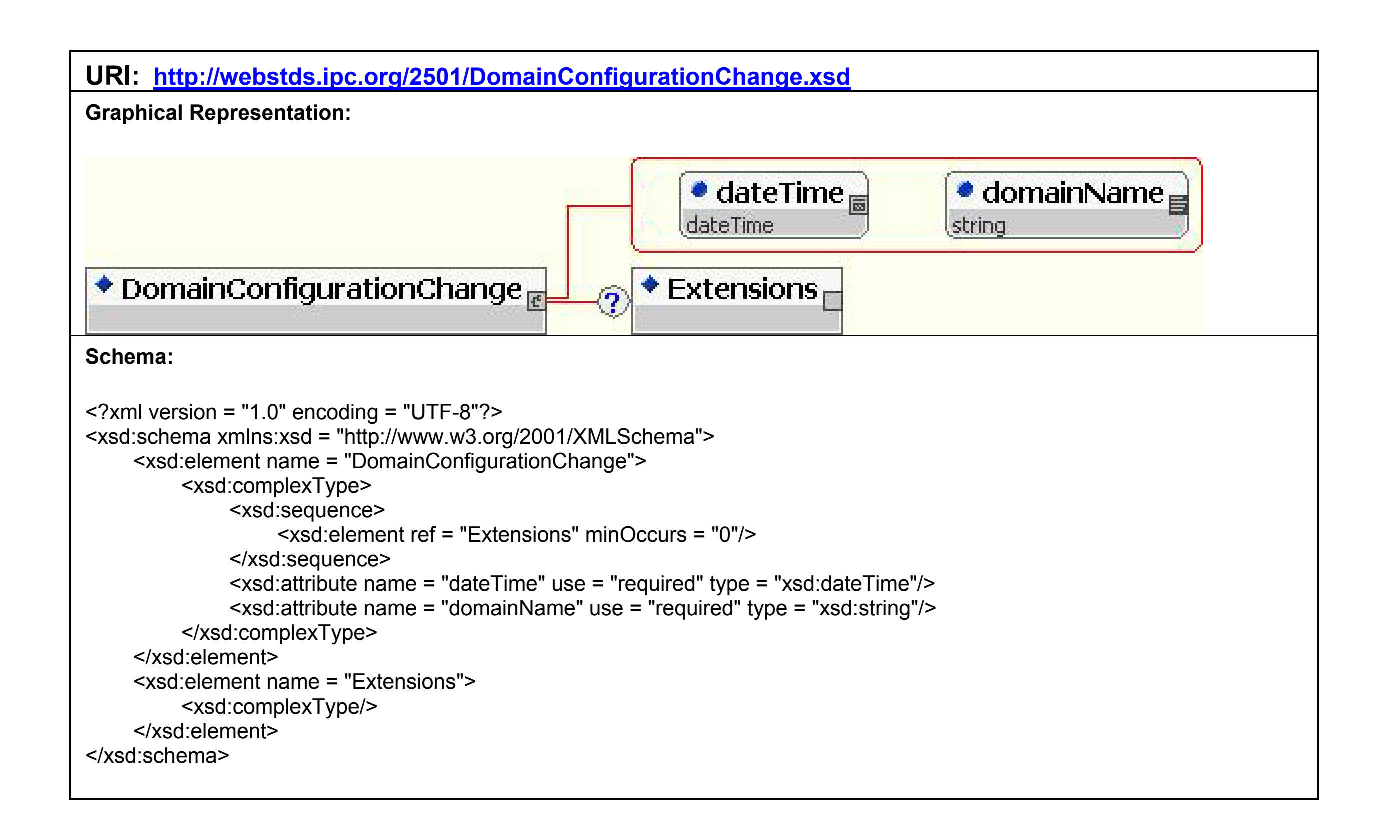
```
            <xsd:attribute name = "comments" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Client">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "PublishList"/>
                <xsd:element ref = "ReceiveList"/>
                <xsd:element ref = "SubscriptionList"/>
            </xsd:sequence>
            <xsd:attribute name = "clientName" use = "required" type = "xsd:anyURI"/>
            <xsd:attribute name = "queueSize" use = "required" type = "xsd:integer"/>
            <xsd:attribute name = "queueFullOperation" use = "required">
                <xsd:simpleType>
                    <xsd:restriction base = "xsd:string">
                        <xsd:enumeration value = "STOP"/>
                        <xsd:enumeration value = "ERASE"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:attribute>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "SubscriptionList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "Publisher" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "PublishList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "MessageSchema" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "ReceiveList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "Sender" minOccurs = "0" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Publisher">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "MessageSchema" maxOccurs = "unbounded"/>
            </xsd:sequence>
            <xsd:attribute name = "publisherName" use = "required" type = "xsd:anyURI"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "MessageSchema" type = "xsd:anyURI"/>
    <xsd:element name = "ClientList">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "Client" maxOccurs = "unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Broker">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "PublishList"/>
            </xsd:sequence>
            <xsd:attribute name = "brokerName" use = "required" type = "xsd:anyURI"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Sender">
        <xsd:complexType>
```

```
            <xsd:sequence>
                <xsd:element ref = "MessageSchema" maxOccurs = "unbounded"/>
            </xsd:sequence>
            <xsd:attribute name = "senderName" use = "required" type = "xsd:anyURI"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

4.3 DomainConfigurationChange

This message indicates that the DomainConfiguration has changed. To obtain the new DomainConfiguration, Clients can send a GetDomainConfiguration to the Message Broker.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC
dateTime	dateTime	The time stamp capturing the instant the Message Broker created the DomainConfigurationChange message.	1
domainName	string	The name of the Domain that the Message Broker is servicing.	1
Extensions	Element	An optional element for containing non-standard XML messages and references.	0-1



URI: http://webstds.ipc.org/2501/DomainConfigurationChange.xsd

Graphical Representation:

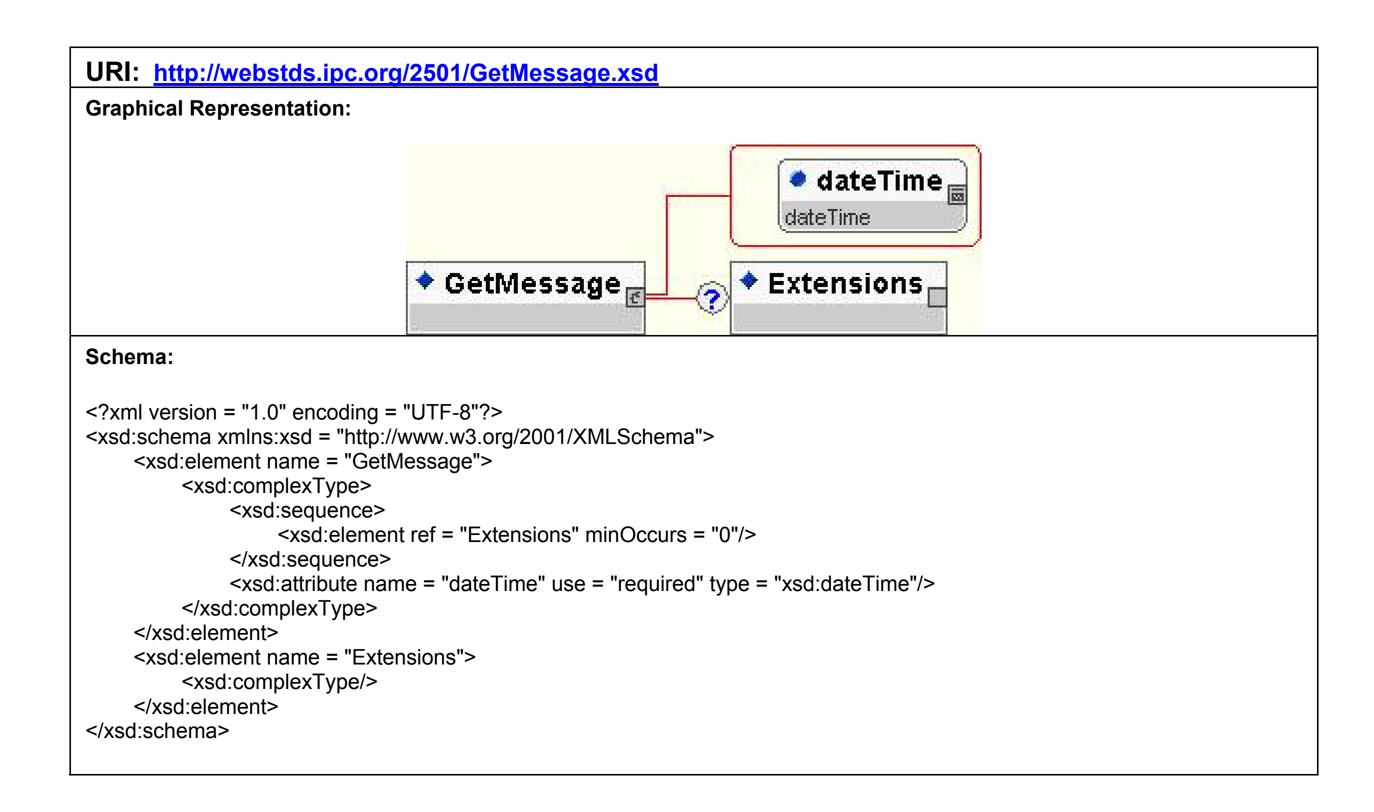
Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
    <xsd:element name = "DomainConfigurationChange">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref = "Extensions" minOccurs = "0"/>
            </xsd:sequence>
            <xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
            <xsd:attribute name = "domainName" use = "required" type = "xsd:string"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name = "Extensions">
        <xsd:complexType/>
    </xsd:element>
</xsd:schema>
```

4.4 GetMessage

Clients send this message to the Message Broker to retrieve queued messages stored by the Message Broker for the Client.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC
dateTime	dateTime	The time stamp capturing the instant the Client created the request for Message Broker-side messages.	1
Extensions	Element	An optional element for containing non-standard XML messages and references.	0-1



URI: http://webstds.ipc.org/2501/GetMessage.xsd

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
	<xsd:element name = "GetMessage">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref = "Extensions" minOccurs = "0"/>
			</xsd:sequence>
			<xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name = "Extensions">
		<xsd:complexType/>
	</xsd:element>
</xsd:schema>
```

4.5 Acknowledge

Acknowledge is used to confirm the receipt of a message.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC
dateTime	dateTime	The time stamp capturing the instant the Message Broker created the DomainCharacteristics message.	1
MessageId	Element	An element representing each messages unique ID.	1
Extensions	Element	An optional element for containing non-standard XML messages and references.	0-1

URI: http://webstds.ipc.org/2501/Acknowledge.xsd

Graphical Representation:

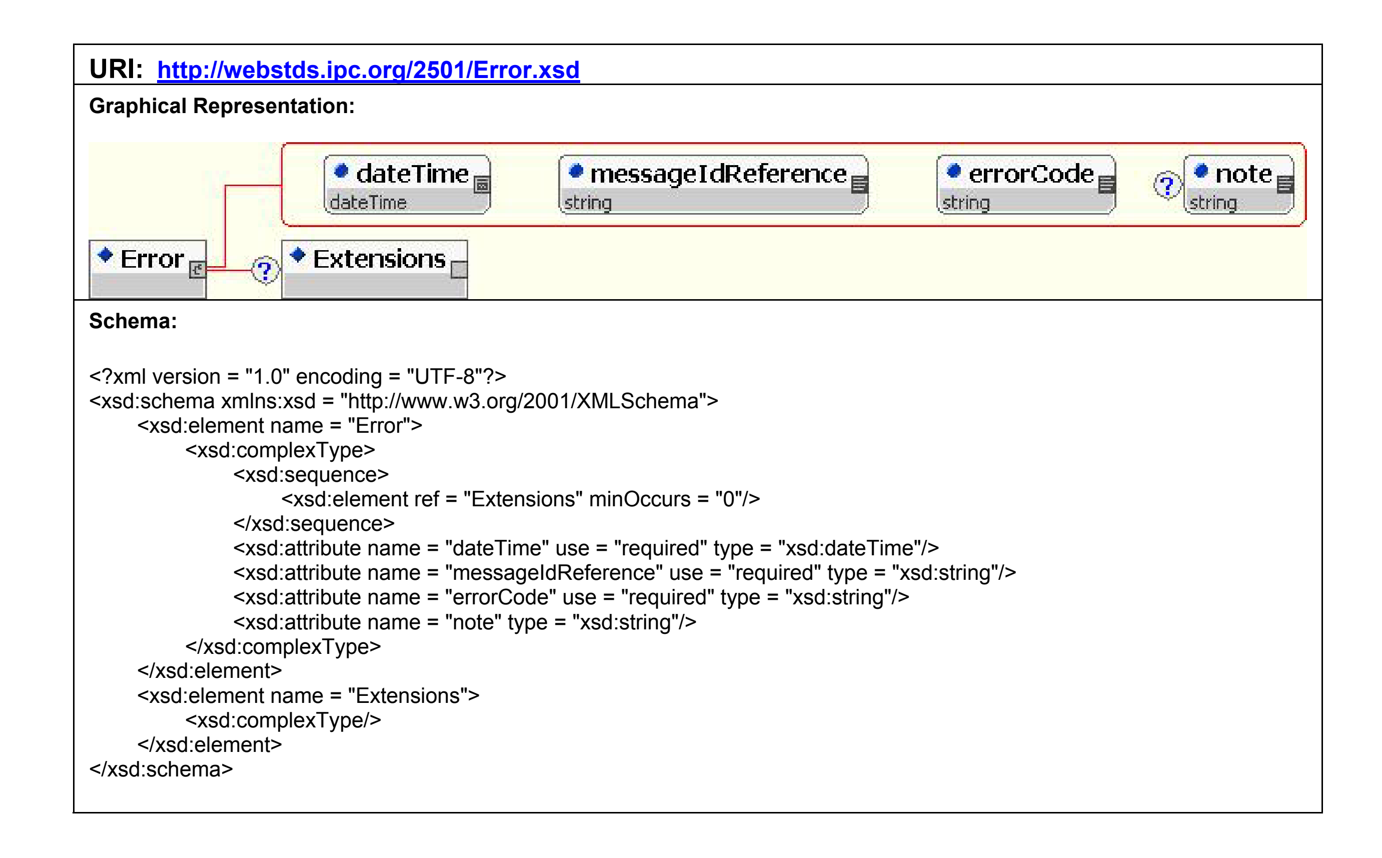
Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
	<xsd:element name = "Acknowledge">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref = "MessageId"/>
				<xsd:element ref = "Extensions" minOccurs = "0"/>
			</xsd:sequence>
			<xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name = "Extensions">
		<xsd:complexType/>
	</xsd:element>
	<xsd:element name = "MessageId" type = "xsd:string"/>
</xsd:schema>
```

4.6 Error

The Error message is used to communicate that an applications error has occurred. It is generated when the receiver of a message (Client or Message Broker) determines that a message is internally inconsistent, or when the receiver is not able to recognize or resolve the message content. A set of predefined errors must be supported by an IPC-2501 compliant Message Broker.

ATTRIBUTE NAME	ATTRIBUTE TYPE	DESCRIPTION	OCC
dateTime	dateTime	The time stamp capturing the instant the Acknowledge message was created.	1
messageIdReference	string	A reference to the messageId of the problem message.	1
errorCode	string	A unique value for each error type.	1
note	string	The description of the error.	0-1
Extensions	Element	An optional element for containing non-standard XML messages and references.	0-1



URI: http://webstds.ipc.org/2501/Error.xsd

Graphical Representation:

Schema:

```
<?xml version = "1.0" encoding = "UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema">
	<xsd:element name = "Error">
		<xsd:complexType>
			<xsd:sequence>
				<xsd:element ref = "Extensions" minOccurs = "0"/>
			</xsd:sequence>
			<xsd:attribute name = "dateTime" use = "required" type = "xsd:dateTime"/>
			<xsd:attribute name = "messageIdReference" use = "required" type = "xsd:string"/>
			<xsd:attribute name = "errorCode" use = "required" type = "xsd:string"/>
			<xsd:attribute name = "note" type = "xsd:string"/>
		</xsd:complexType>
	</xsd:element>
	<xsd:element name = "Extensions">
		<xsd:complexType/>
	</xsd:element>
</xsd:schema>
```

Predefined Error Codes and Notes

errorCode	NOTE
NO ATTACHMENTS INCLUDED	No attachments were included with request.
INVALID ENTITY NAME	Your name is not a valid entity in the Domain.
NOT WELL FORMED XML 2501	The 2501 message transmitted was not well formed XML.
NOT VALID XML 2501	The 2501 message transmitted was not valid XML.
TYPE MISMATCH	Message sent was not of the type described in the message header.
UNRECOGNIZED MESSAGE TYPE 2501	Unrecognized 2501 message type transmitted.
NO MESSAGE TO ACKNOWLEDGE	You are attempting to acknowledge a message that was not sent to you, has expired in transit, or should not have been acknowledged.
BAD DOMAIN	This broker is not servicing the requested Domain.
NO PERMISSION TO PUBLISH	The current Domain configuration does not grant you permission to publish.
NO PUBLISHING SCHEMA IN DOMAIN	The current Domain configuration does not grant you permission to publish messages of this schema.
MESSAGEID NOT UNIQUE	Duplicate (non-unique) messageId used to identify a message referencing a different message schema. Message will not be processed..
UNNECESSARY PUBLISHING	It is not necessary for you to send messages of this schema because no Clients in this Domain have subscribed to the selected schema.
QUEUE FULL	Your queue has become full; as a result, some messages may have been truncated.
BAD POINT TO POINT DESTINATION	The destination Client was not found in the Domain
NO POINT TO POINT PERMISSION IN DOMAIN	The current Domain configuration does not grant permission to send a message of that schema to that Client

5 Process Flow Diagrams.

Section 3.3 outlines the simple choreography of successful message transfer. This section contains non-normative process flow diagrams outlining the decisions to be made by lower level applications, to assist in the development of IPC-2501 Clients.

5.1 Client to Message Broker Transfer

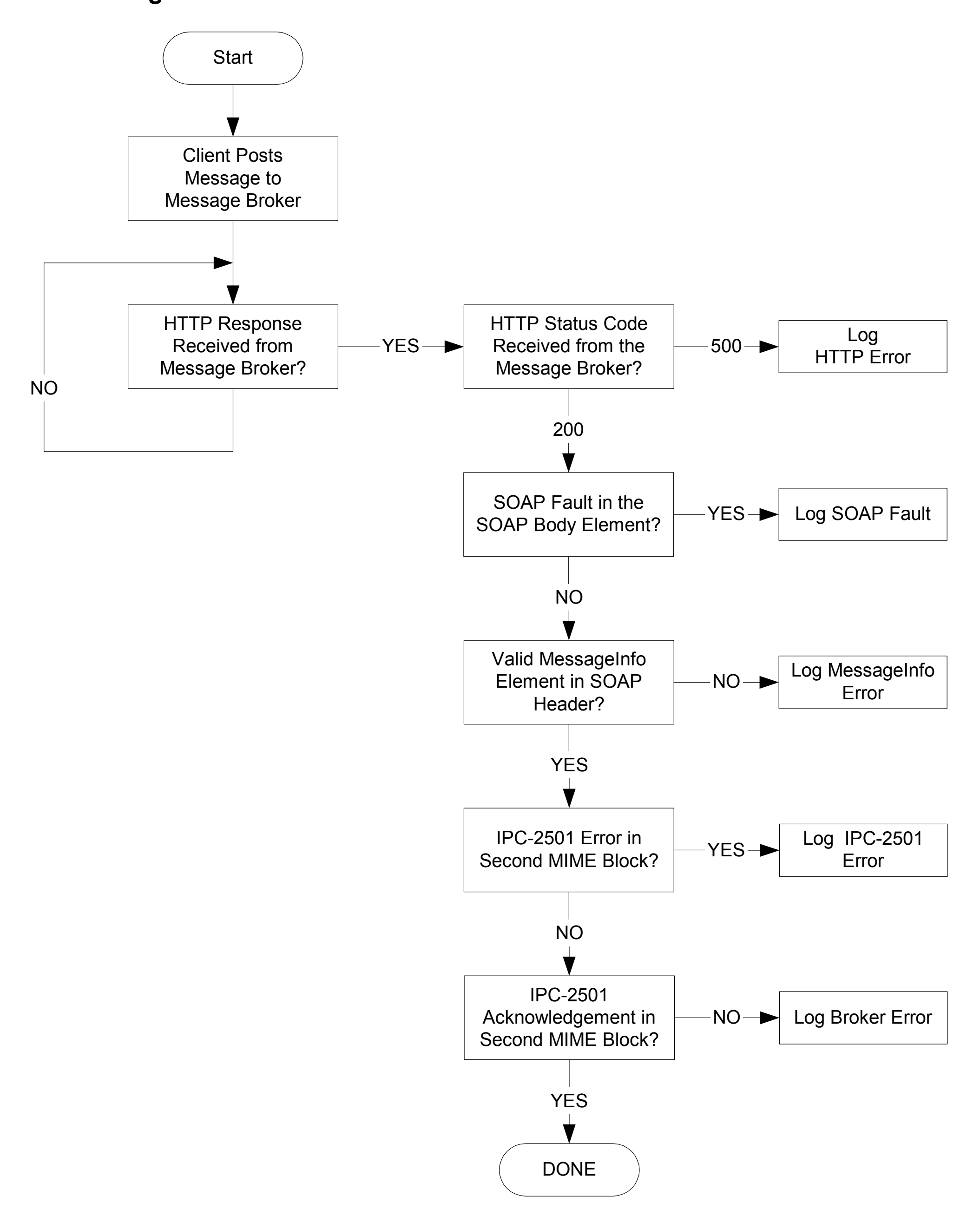


Figure 10 The process flow diagram for a Client transferring a message to a Message Broker

5.2 Message Broker to Client Transfer

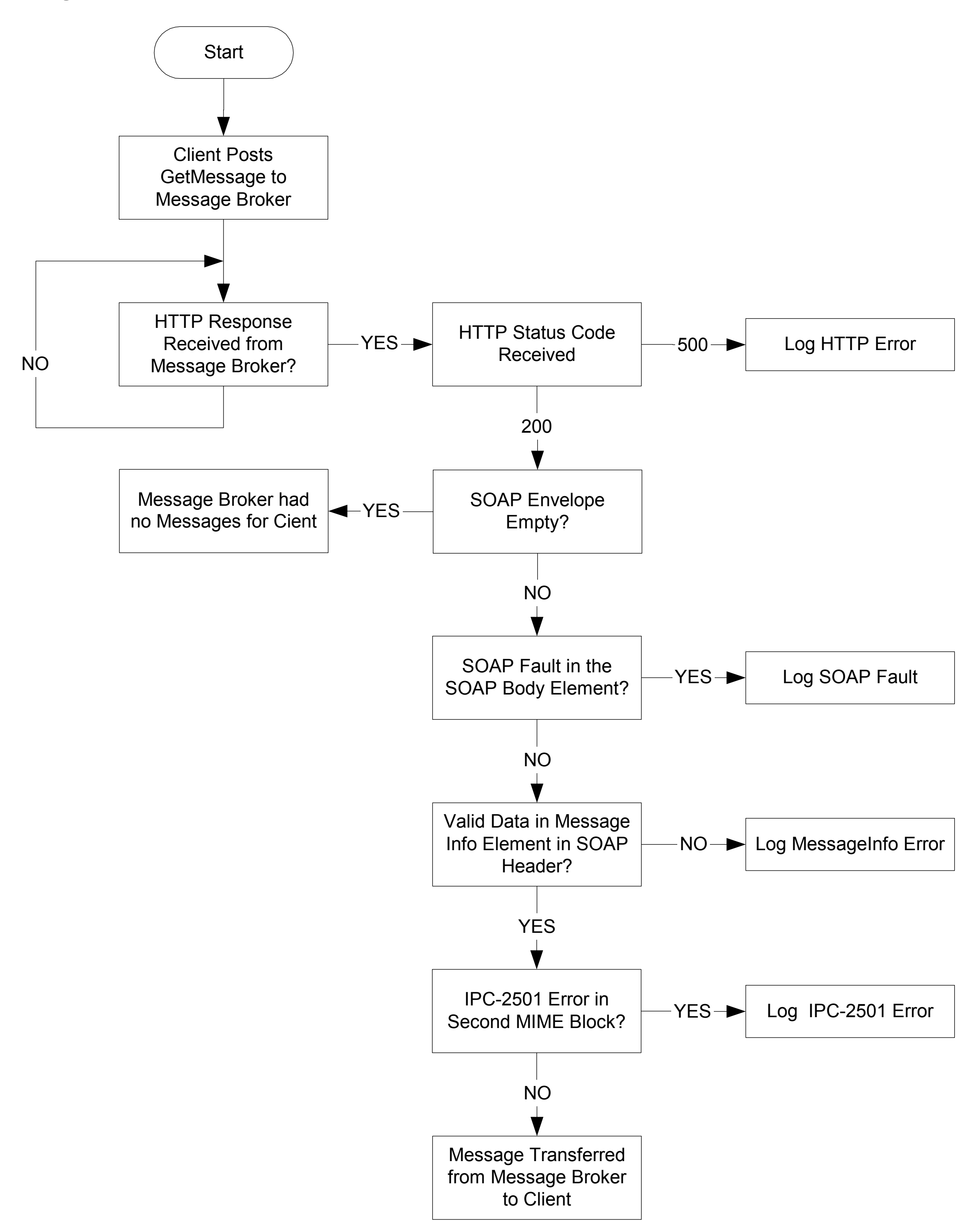


Figure 11 The process flow diagram for a Client receiving a message from the Message Broker

6 Example Domain Configuration

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<IPC2501DC:DomainConfiguration
    xmlns:IPC2501DC="http://webstds.gatech.edu/2501/DomainConfiguration.xsd"
    dateTime="2003-03-27T15:54:17.02-05:00"
    comments="New clients for IPC-2501 doc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    author="Doug"
    xsi:noNamespaceSchemaLocation="http://webstds.gatech.edu/2501/DomainConfiguration.xsd"
    domainName="IPC-2501 Reference Implementation">
<Broker brokerName="broker.fis.marc.gatech.edu">
<PublishList>

    <MessageSchema>http://webstds.gatech.edu/2501/DomainConfigurationChange.xsd</Message
    Schema>
   </PublishList>
   </Broker>
<ClientList>
<Client clientName="Client1.marc.gatech.edu" queueSize="200" queueFullOperation="STOP">
<PublishList>
  <MessageSchema>http://webstds.gatech.edu/2501/GetDomainConfiguration.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarm.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentChangeState.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentError.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentHeartbeat.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentWarning.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemIdentifierRead.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferIn.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferLane.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferOut.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferZone.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkStart.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkComplete.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/OperatorInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2546/MaterialHandlerTrouble.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionStart.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ItemProcessStatus.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessStepStatus.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ItemRepair.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/InspectionFrame.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionEnd.xsd</MessageSchema>
</PublishList>
<ReceiveList>
<Sender senderName="Client2.marc.gatech.edu">
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarmCleared.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/OperatorInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentPowerOff.xsd</MessageSchema>
</Sender>
</ReceiveList>
<SubscriptionList>
<Publisher publisherName="broker.fis.marc.gatech.edu">
<MessageSchema>http://webstds.gatech.edu/2501/DomainConfigurationChange.xsd</MessageSchema>
</Publisher>
<Publisher publisherName="Client2.marc.gatech.edu">
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarm.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentChangeState.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentError.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentHeartbeat.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentWarning.xsd</MessageSchema>
```

```
  <MessageSchema>http://webstds.ipc.org/2541/ItemIdentifierRead.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferIn.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferLane.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferOut.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferZone.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkStart.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkComplete.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarmCleared.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentPowerOff.xsd</MessageSchema>


   </Publisher>
   </SubscriptionList>
   </Client>
<Client clientName="Client2.marc.gatech.edu" queueSize="200" queueFullOperation="STOP">
<PublishList>
   <MessageSchema>http://webstds.gatech.edu/2501/GetDomainConfiguration.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarm.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentChangeState.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentError.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentHeartbeat.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentWarning.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemIdentifierRead.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferIn.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferLane.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferOut.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemTransferZone.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkStart.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/ItemWorkComplete.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/OperatorInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2546/MaterialHandlerTrouble.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionStart.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ItemProcessStatus.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessStepStatus.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/InspectionFrame.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionEnd.xsd</MessageSchema>
</PublishList>
<ReceiveList>
<Sender senderName="Client1.marc.gatech.edu">
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentWarning.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/OperatorInformation.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarmCleared.xsd</MessageSchema>
  <MessageSchema>http://webstds.ipc.org/2541/EquipmentPowerOff.xsd</MessageSchema>

</Sender>
</ReceiveList>
<SubscriptionList>
<Publisher publisherName="broker.fis.marc.gatech.edu">
 <MessageSchema>http://webstds.gatech.edu/2501/DomainConfigurationChange.xsd</MessageSche
     ma>
</Publisher>
<Publisher publisherName="Client1.marc.gatech.edu">
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarm.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentChangeState.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentError.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentHeartbeat.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentInformation.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentWarning.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemIdentifierRead.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemTransferIn.xsd</MessageSchema>
```

```
    <MessageSchema>http://webstds.ipc.org/2541/ItemTransferLane.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemTransferOut.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemTransferZone.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemWorkStart.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/ItemWorkComplete.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentAlarmCleared.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2541/EquipmentPowerOff.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2546/MaterialHandlerTrouble.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionStart.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2547/ItemProcessStatus.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2547/ProcessStepStatus.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2547/InspectionFrame.xsd</MessageSchema>
    <MessageSchema>http://webstds.ipc.org/2547/ProcessSessionEnd.xsd</MessageSchema>
  </Publisher>
  </SubscriptionList>
</Client>
</ClientList>
</IPC2501DC:DomainConfiguration>
```

7 References

HTTP – Hypertext Transport Protocol is an application-level, generic, stateless, object-oriented protocol for distributed, collaborative, hypermedia information systems. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP has been in use by the World-Wide Web global information initiative since 1990. For further information about HTTP 1.1, see http://www.w3.org/Protocols/rfc2616/rfc2616.html.

MIME - Multipurpose Internet Mail Extension. MIME was conceived to extend the format of Internet mail to allow non-US-ASCII textual messages, non-textual messages, multipart message bodies, and non-US-ASCII information. See RFC2045, RFC2046, RFC2047, RFC2048, RFC2049 for full specifications.

SOAP – Simple Object Access Protocol. A transport-independent protocol that uses XML to invoke remote methods.

For more information about SOAP Version 1.1, see:
http://www.w3.org/TR/SOAP/

For more information about SOAP with Attachments, see:
http://www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211.html

The SOAP 1.1 envelope schema can be found at:
http://schemas.xmlsoap.org/soap/envelope/

TCP/IP – An Internet communications protocol comprised of two components. TCP is responsible for verifying the correct delivery of data from client to server, and IP is responsible for moving packet of data from node to node.

XML – eXtensible Markup Language. A meta-language (based on the ISO 8879 standard on Standard Generalized Markup Language) for describing embedded markup languages with particular emphasis on the World Wide Web Communications Architecture.

Appendix A – IPC Web-based Standards (IPC25XX)

The web-based standards (IPC 25XX) are designed to foster application integration and electronic commerce through data and information interchange standards based on XML. It assumes that application programs (including equipment interfaces) are distinct entities, and application integration takes place using a loosely coupled, message-passing approach. There is no need for a common object model, programming language, persistent storage mechanism or operating system for two applications to exchange XML messages formatted using the web-based standards. The two applications simply need to be able to format, transmit, receive and consume a standardized XML message.

The IPC web-based standards series have been identified for each of the value-added activities occurring throughout the product life cycle of an electronics product. The web-based standards are:

IPC-2500 – Framework Standard

IPC-2510 – Product Data Representation

IPC-2520 – Product Data Quality

IPC-2530 – Surface Mount Equipment Standard Recipe File Format

IPC-2540 – Shop Floor Equipment Communications

IPC-2550 – Manufacturing Execution Systems Communications

IPC-2560 – Enterprise Resource Planning Systems Communications

IPC-2570 – Supply Chain Communications

IPC-2580 – Product Manufacturing Descriptions

Table A-1 shows the correlation of the different standards in each of the series. Although not every standard has been started, the figure represents a coordinated opportunity to maintain consistency throughout the standard development cycle.

Table A-1 CAD/CAM Standardization

IPC Number/ Function	-xxx1 Generic	-xxx2 Administ	-xxx3 Documnt	-xxx4 Board Fabricat	-xxx5 Bare Bd Test	-xxx6 Assy Manufac	-xxx7 Assy/ Test/ Insp.	-xxx8 Comp. & Material	-xxx9 Informa. Modeling
IPC-2500 CAMX Framework	IPC-2501 (Pub)								
IPC-2510 GenCAM Product Data	IPC-2511A 2511B (Pub)	IPC-2512A (Pub)	IPC-2513A (Pub)	IPC-2514A (Pub)	IPC-2515A (Pub)	IPC-2516A (Pub)	IPC-2517A (Pub)	IPC-2518A (Pub)	IPC-2519A (Pub)
IPC-2520 Quality Product Data				IPC-2524 (Pub)					
IPC-2530 SRFF Process Data Recipe file	IPC-2531 (Pub)								
IPC-2540 Shop Floor Communication	IPC-2541 (Pub)					IPC-2546 (Pub)	IPC-2547 (Pub)		
IPC-2550 Execution Communication	IPC-2551 Working draft			IPC-2554 Working draft					
IPC-2560 Enterprise Communication									
IPC-2570 Supply Chain Communication	IPC-2571 (Pub)					IPC-2576 (Pub)	IPC-2577 (Final draft)	IPC-2678 (Pub)	
IPC-2580 Product Manufacturing Descriptions	IPC-2581 Propsd Std								

Messages are the basis of the web-based standards. Messages are the means to integrate applications at the business-process level by defining a loosely coupled, request-based communication process. Since many business processes involve one party performing a service at the request of another party, the mapping of messages to requests is natural. An XML-based messaging system with open, extensible formats captures the essential elements of an electronics business communication message while allowing flexible implementations.

It is anticipated that in the vast majority of interchanges, the exchange of XML documents and messages between trading partners or applications will occur. Implementation using the web-based standards will use a simple hyper-text transfer protocol (HTTP) transport, but business can also use other transports including file transfer protocol (FTP) and message queuing technologies. At times, a message may be short and distinct; other times the message may contain a large file or a linkage to a URI. In many instances, the data represents an action required and becomes a part of a contractual agreement between customer and supplier.

In today's environment, many new applications are gaining native support for XML schema. These message and file transfers will require layered software that transforms native data types into XML or vice versa and converts the characteristics of the XML message into processing actions by machines, personnel, or processes.